Tiva Based Daughter Board for Firebird V Hardware And Software Manual.

eRTS Lab IIT Bombay

July 4, 2017

1 Credits

Version 1.0
July 4, 2017

Documentation Author(Alphabetical Order):

1. Ayush Gaurav, Intern eYSIP 2017
2. Nagesh K, Intern eYSIP 2017

Credits(Alphabetical Order):

1. Prof Kavi Arya, CSE IIT Bombay
2. Nex Robotics Pvt. Ltd.
3. Piyush Manavar, Team e-Yantra
4. Saurav Shandilya, Team e-Yantra

2 Notice

The contents of this manual are subject to change without notice. All efforts have been made to ensure the accuracy of contents in this manual. However, should any errors be detected, e-yantra welcomes your corrections. You can send us your queries / suggestions at Contact Us

Table Of Content

3 Introduction

Tiva Daughter board for Fire Bird V will help you gain exposure to the world of robotics and embedded systems with ARM Cortex M4. The board is designed with Open Source Philosophy in software and hardware design ,you will be able to create and contribute to complex applications that run on this platform, helping you acquire expertise as you spend more time with them.

3.1 Safety precautions:

- Robot's electronics is static sensitive. Use robot in static free environment.
- Read the assembling and operating instructions before working with the robot.
- If robot's battery low buzzer starts beeping, immediately charge the batteries.
- To prevent fire hazard, do not expose the equipment to rain or moisture.
- Refrain from dismantling the unit or any of its accessories once robot is assembled.
- Charge the NiMH battery only with the charger provided on the robot.
- Never allow NiMH battery to deep discharge.
- Mount all the components with correct polarity.
- Keep wheels away from long hair or fur.
- Keep the robot away from the wet areas. Contact with water will damage the robot.
- To avoid risk of fall, keep your robot in a stable position.
- Do not attach any connectors while robot is powered ON.
- Never leave the robot powered ON when it is not in use.
- Disconnect the battery charger after charging the robot.

3.2 Inappropriate Operation:

Inappropriate operation can damage your robot. Inappropriate operation includes, but is not limited to:

- Dropping the robot, running it off an edge, or otherwise operating it in irresponsible manner.
- Interfacing new hardware without considering compatibility.
- Overloading the robot above its payload capacity.
- Exposing the robot to wet environments.
- Continuing to run the robot after hair, yarn, string, or any other item is entangled in the robot's axles or wheels.
- All other forms of inappropriate operations.
- Using robot in areas prone to static electricity.
- Read carefully paragraphs marked with caution symbol.

4 Tiva Based Daughter Board

There are two daughter boards one with the launchpad and other one with the Arm Cortex M4 based uC. Almost all the specification are same unless mentioned otherwise.

4.1 Technical Specification

Microcontroller:
TM4C123gh6pm (ARM architecture based Microcontroller)
To know more about the microcontroller please refer to datasheet.

Sensors:
Three white line sensors (extendable to 7)
Five Sharp GP2Y0A02YK IR range sensor (One in default configuration)
Eight analog IR proximity sensors
Two position encoders

Indicators:
2 x 16 Characters LCD
Buzzer

Communication:
USB Communication
Wireless ZigBee Communication (2.4GHZ) (if XBee wireless module is installed)
Bluetooth communication(Can be interfaced on external UART0 available on the board)
Simplex infrared communication (From infrared remote to robot)
I2C Communication

Battery Life:
2 Hours, while motors are operational at 75% of time

Locomotion:
Two DC geared motors in differential drive configuration and caster wheel at front as support
Top Speed: 24 cm / second
Wheel Diameter: 51mm
Position encoder: 30 pulses per revolution
Position encoder resolution: 5.44 mm

5 Software Manual

5.1 Simple I/O Operation

In this section, we will learn how to initialize GPIO pins in TM4C123G and use them for simple input-output operations. The code for TIVA is written in C99 programming standards with Hungarian standard for naming variables.

5.1.1 Header Files

Before writing the code we have to include certain header files to access the TivaWare API's. These header files are required for variable definition. The header files to included are given below. The purpose of each header file is explained in the following paragraphs.

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
```

stdint.h: This is used for defining variables according to C99 Standard.
stdbool.h: This is used for defining variables according to C99 Standard.
hw_memmap.h: This contains the marcos defining the memory mapping of Tiva C Series devices. This includes base address locations of all the ports present in Tiva C Series devices. For example, the base address of port B can be accessed by "GPIO_PORTB_BASE".
hw_types.h: This header file includes macros for initialising a pin as input or output. For example, "GPIOPinTypeGPIOOutput" is used for defining a pin as an output pin. For example, "GPIOPinTypeGPIOOutput" is used for defining a pin as an output pin.
sysctl.h : This contains functions and macros for System Control API of DriverLib.
gpio.h: This contains functions and macros for GPIO Control API of DriverLib.

5.1.2 Clock Setup

The TM4C123G contains 2 external clocks. The main oscillator contains a 16 MHz crystal and the Real Time Clock(RTC) contains a 32.768 MHz crystal. The TM4C123G can be configured to operate at different frequencies. In this case, we will configure the clock to operate at 40 MHz.

```
SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
```

The 16MHz crystal drives the 400MHz PLL. There is default /2 divider in the path. We are further dividing this by 5, which means that the 400MHz PLL is divided by a total factor of 10. Hence the System Clock is set at 40MHz.
Refer to the datasheet to configure the clock at different frequencies.

5.1.3 GPIO Configuration

The GPIO pins present in TM4C123G can be configured as input, output and interrupt. The clock of TM4C123G has to enabled for a port before using it as an input or output. This can be done by the following line of code.

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB)
```

In this case, clock is enabled for port B. After enabling the clock, we have to define the status of the pin. Consider an example where pin B6 should be defined as an output pin. This can be done by the following statement.

```
GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_6)
```

Now we define the same pin(B6) as an input pin. This can be done by the following statement.

```
GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_6)
```

But this not sufficient to use it as an input pin. We have to enable the internal pull-up or pull-down resistor associated with a pin. This is shown below

```
GPIOPadConfigSet(GPIO_PORTB_BASE, GPIO_PIN_6 ,GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU)
```

The internal pull-down resistor can be enabled by replacing WPU(Weak Pull Up) by WPD(Weak Pull Down).

5.1.4 Assignments

- The buzzer is connected to PF4 in Plug and Play board. Write a code to switch the buzzer on and off alternatively.
- The buzzer is connected to PA2 in uC based board. Write a code to switch the buzzer on and off alternatively.
- The motor connections are as shown below. Write a code to move the bot in forward, backward, left and right directions.

Motor Pin	**Pin Name(uC)**	**Pin Name(Plug and Play)**
L1	PB0	PF3
L2	PB1	PB3
PWM Left	PF3	PF2
R1	PF4	PC4
R2	PA5	PC6
PWM Right	PA6	PC5